paexec – distributes tasks over network or CPUs

Aleksey Cheusov
vle@gmx.net

Minsk, Belarus, 2011

Problem

- Huge amount of data to process
- Typical desktop machines have more than one CPU/Core
- Unlimited resources are available through Internet
- Heterogeneous environment (*BSD, Linux, Windows...)

Solution

Usage

```
paexec [OPTIONS] \
  -n 'machines of CPUs' \
  -t 'transport program' \
  -c 'calculator' < tasks
```

example

```
ls *.wav | paexec \
  -n +4 \
  -c ~/bin/wav2flac
```

example

```
paexec \
  -n 'host1 host2 host3' \
  -t /usr/bin/ssh \
  -c ~/bin/toupper < tasks
```

Example 1: toupper

Our task is to convert strings to upper case

~/bin/toupper

```
#!/usr/bin/awk -f

{
   print " ", toupper($0)
   print ""  # end-of-task marker!
   fflush()  # We must flush stdout!
}
```

~/tmp/tasks

```
apple
bananas
orange
```

Example 1: toupper

paexec invocation

```
$ paexec -t ssh -c ~/bin/toupper \
   -n 'syn-proc7 syn-proc5' \
   < ~/tmp/tasks > results
$ cat results
 BANANAS
 ORANGE
 APPLE
$
```

Example 1: toupper

paexec -lr invocation

```
$ paexec -lr -t ssh -c ~/bin/toupper \
   -n 'syn-proc7 syn-proc5' \
   < ~/tmp/tasks > results
$ cat results
syn-proc5 2  BANANAS
syn-proc5 3  ORANGE
syn-proc7 1  APPLE
$
```

Example 2: parallel banner(1)

what is banner(1)?

```
$ banner -f @ NetBSD

@      @                        @@@@@@    @@@@@  @@@@@@
@@     @   @@@@@@    @@@@@ @      @ @      @ @      @
@ @    @   @         @    @      @ @        @      @
@  @   @   @@@@@     @    @@@@@@   @@@@@  @      @
@   @  @   @         @    @      @        @ @      @
@    @@   @         @    @      @ @      @ @      @
@     @   @@@@@@    @    @@@@@@   @@@@@  @@@@@@

$
```

Example 2: parallel banner(1)

~/bin/pbanner

```
#!/bin/sh

while read task; do
   banner -f M "$task" | sed 's/^/ /'
   echo '' # end-of-task-marker
done
```

~/bin/tasks2

```
pae
xec
```

paexec invocation

```
$ paexec -l -c ~/bin/pbanner \
   -n +2 \
   < ~/tmp/tasks2 > results
$
```

Example 2: parallel banner(1)

Sliced result

```
$ cat results
2
2
2   M     M   MMMMMM    MMMM
2    M   M    M         M     M
1
1
1   MMMMM       MM      MMMMMM
1   M     M    M  M     M
1   M     M   M     M   MMMMM
1   MMMMM    MMMMMM    M
2      MM    MMMMM     M
2      MM    M         M
2    M   M   M         M     M
2   M     M   MMMMMM    MMMM
2
1   M         M     M   M
1   M         M     M   MMMMMM
1
$
```

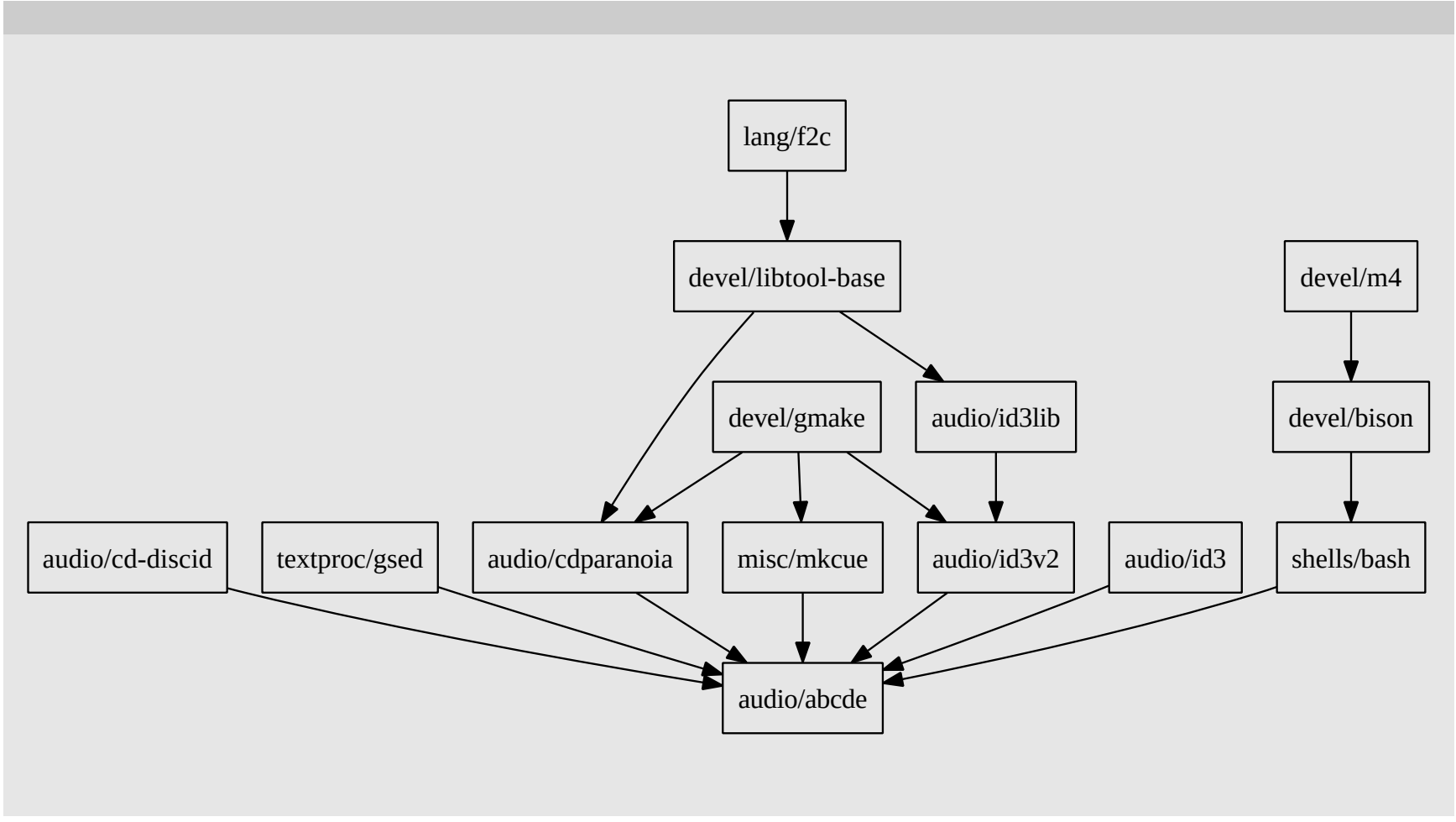
Example 2: parallel banner(1)

Ordered result

```
$ paexec_reorder -S results

 MMMMM      MM     MMMMMM
 M    M    M  M    M
 M    M   M    M   MMMMM
 MMMMM    MMMMMM   M
 M        M    M   M
 M        M    M   MMMMMM


 M    M   MMMMMM    MMMM
  M  M    M        M    M
   MM     MMMMM    M
   MM     M        M
  M  M    M        M    M
 M    M   MMMMMM    MMMM

$
```

Example 3: dependency graph of tasks

paexec(1) is able to build tasks taking into account their "dependencies"



Example 3: dependency graph of tasks

~/tmp/packages_to_build

```
audio/cd-discid audio/abcde
textproc/gsed audio/abcde
audio/cdparanoia audio/abcde
audio/id3v2 audio/abcde
audio/id3 audio/abcde
misc/mkcue audio/abcde
shells/bash audio/abcde
devel/libtool-base audio/cdparanoia
devel/gmake audio/cdparanoia
devel/libtool-base audio/id3lib
devel/gmake audio/id3v2
audio/id3lib audio/id3v2
devel/m4 devel/bison
lang/f2c devel/libtool-base
devel/gmake misc/mkcue
devel/bison shells/bash
```

Example 3: dependency graph of tasks

~/bin/pkg_builder

```
#!/usr/bin/awk -f

{
   print "build " $0
   print "success" # build succeeded!
   print ""         # end-of-task marker
   fflush()         # we must flush stdout
}
```

Example 3: dependency graph of tasks

paexec -g invocation (no failures)

```
$ paexec -g -l -c ~/bin/pkg_builder -n 'syn-proc5 syn-proc7' \
    -t ssh < ~/tmp/packages_to_build | paexec_reorder > result
$ cat result
build textproc/gsed
success
build devel/gmake
success
build misc/mkcue
success
build devel/m4
success
build devel/bison
success
...
build audio/id3v2
success
build audio/abcde
success
$
```

Example 3: dependency graph of tasks

~/bin/pkg_builder

```
#!/usr/bin/awk -f

{
   print "build " $0
   if ($0 == "devel/gmake")
      print "failure" # Oh no...
   else
      print "success" # build succeeded!

   print ""         # end-of-task marker
   fflush()         # we must flush stdout
}
```

Example 3: dependency graph of tasks

paexec -g invocation (with failures)

```
$ paexec -gl -c ~/bin/pkg_builder -n 'syn-proc5 syn-proc7' \
    -t ssh < ~/tmp/packages_to_build | paexec_reorder > result
$ cat result
build audio/cd-discid
success
build audio/id3
success
build devel/gmake
failure
devel/gmake audio/cdparanoia audio/abcde audio/id3v2 misc/mkcue
build devel/m4
success
build textproc/gsed
success
...
$
```

Example 4: Resistance to network failures

~/bin/pkg_builder

```
#!/usr/bin/awk -f

{
   "hostname -s" | getline hostname
   print "build " $0 " on " hostname

   if (hostname == "syn-proc7" && $0 == "textproc/gsed")
      exit 0 # Damn it, I'm dying...
   else
      print "success" # Yes! :-)

   print ""         # end-of-task marker
   fflush()         # we must flush stdout
}
```

Example 4: Resistance to network failures

paexec -Z300 invocation (with failure)

```
$ paexec -gl -Z300 -t ssh -c ~/bin/pkg_builder \
      -n 'syn-proc5 syn-proc7' < ~/tmp/packages_to_build \
      | paexec_reorder > result
$ cat result
build audio/cd-discid on syn-proc5
success
build textproc/gsed on syn-proc7
fatal
build textproc/gsed on syn-proc5
success
build audio/id3 on syn-proc5
success
...
$
```

The End